PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-132502
(43)Date of publication of application : 10.05.2002

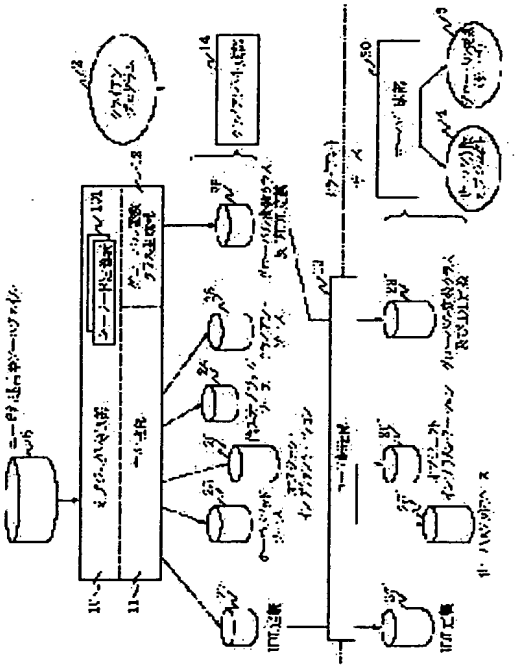
(51)Int.Cl. G06F 9/44
G06F 9/45
G06F 15/16

(21)Application number : 2000-325652
(22)Date of filing : 25.10.2000

(71)Applicant : NEC SOFTWARE HOKURIKU LTD
(72)Inventor : NODA MASAHIKO

(54) AUTOMATIC CREATING SYSTEM AND METHOD OF DISTRIBUTED OBJECT WITH INTERPRETATION OF LANGUAGE FUNCTION

(57)Abstract:
PROBLEM TO BE SOLVED: To eliminate the need for a decision of a performing position, for IDL description of a server object, for a marshalling, for an un-marshalling process coding and the like operated conventionally by users to create distributed processing program from a user description language.
SOLUTION: A client program creating part 1 comprises an object dividing part 10 to extract a class from a user description language source 6 for the decision of the performing position, a main creating part 11 to create an IDL definition 27, an implementation 26, a method 25 and a proxy object 24 for creation of a client source 23 with substituting access to the server of the part except the server object of a source 6 for a proxy, a client creating part 14 to perform a compile link of a source 23, of the object 24 and of the IDL definition 27 and a code transferring part 13, and a server program creating part 3 has a server creating means to perform the compile link of the received source.



LEGAL STATUS

[Date of request for examination] 19.09.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

decision of rejection]
[Date of extinction of right]

* NOTICES *

Machine-generated English translation for JP 2002-132502

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] It is the system which generates automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation means and a server program generation means to generate a distributed object are included. Said client program generation means An object division means to extract class processing from said source code, and to distinguish the activation location using a keyword, About the object judged to be server activation, an IDL definition, the object implementation source, The server method source is created. While generating the deputy class source which corresponds to an object implementation and is prepared in a client side A main generation means to use the non-extracting part and client activation class of the source of said user description language as a main source, to permute the processing to the server class of a main source by the access processing to said deputy class, and to generate the client source, A client generation means to consider said client source, the deputy class source, and an IDL definition as an input, to compile and link these, and to generate an execute-form client program, It has a code transfer means to transmit said IDL definition, the object implementation source, and the method source to a server. Said server program generation means The distributed object automatic generative system characterized by having Cong Pal and a server generation means to link and to generate an execute-form server program for a means to receive the transmitted code, and the received code.
[Claim 2] It is the system which generates automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation means and a server program generation means to generate a distributed object are included. Said client program generation means An object division means to extract class processing from said source code, and to distinguish the activation location using a keyword, A global variable class generation means to generate the global variable deputy class which extracts a global variable from said source code and object division information, corresponds to the global variable object and this which manage these IDL definition and these, and is prepared in a client, About the object judged by said division to be server activation, an IDL definition, The object implementation source and the server method source are created. While creating the deputy class source which corresponds to an object implementation and is prepared in a client side, the non-extracting part and client activation class of the source of said user description language are used as a main source. The processing to the server class of a main source is permuted by the access processing to said deputy class. A main generation means to permute the reference to a global variable, and substitution processing by the access processing to said global variable deputy class, and to generate the client source, Said client source, the deputy class source, an IDL definition, a global variable deputy class, A client generation means to consider a global variable IDL definition as an input, to compile and link these, and to generate an execute-form client program, It has a code transfer means to transmit said IDL definition, the object implementation source, the method source, a global variable class, and its IDL definition to a server. Said server program generation means is a distributed object automatic generative system characterized by having a server generation

means to compile and link a means to receive said transmitted code, and the received code, and to generate an execute-form server program.
[Claim 3] As for said client program generation means, said deputy class is called from the client source. The set of a carrier beam argument The processing changed into the format which an object request broker defines is included. Generate said deputy class and the processing said whose object implementation changes the set of a carrier beam argument into the format of a server method program from an object request broker is included. Claim 1 characterized by generating said object implementation source, or a distributed object automatic generative system given in two.
[Claim 4] Said client program generation means said global variable deputy class The processing changed into the format that it is called from the client source and an object request broker defines the set of a carrier beam argument is included. Generate said global variable deputy class and the processing said whose global variable class changes the set of a carrier beam argument into the format of a server method program from an object request broker is included. The distributed object automatic generative system according to claim 2 characterized by generating said global variable class.
[Claim 5] They are claim 1 which said object division means is equipped with keyword information for every class of the corresponding to two or more kinds of user description languages, and is characterized by each keyword information having the logical expression between one or more keyword, and an activation tab-control-specification value at the time of logic formation, or a distributed object automatic generative system given in two.
[Claim 6] Said object division means is claim 1 characterized by to have a means decide an activation location about said extracted class processing according to the explicit activation tab-control-specification information first marked on the source of a user description language, and a means judge the activation location of the class processing without explicit activation tab control specification using said keyword information, or a distributed object automatic generative system given in two.
[Claim 7] It is the approach of generating automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation procedure and the server program generation procedure which generates a distributed object are included. Said client program generation procedure The object division procedure which extracts class processing from said source code, and distinguishes the activation location using a keyword, About the object judged to be server activation, an IDL definition, the object implementation source, The server method source is created. While generating the deputy class source which corresponds to an object implementation and is prepared in a client side The main generation procedure which uses the non-extracting part and client activation class of the source of said user description language as a main source, permutes the processing to the server class of a main source by the access processing to said deputy class, and generates the client source, The client generation procedure which considers said client source, the deputy class source, and an IDL definition as an input, compiles and links these, and generates an execute-form client program, It has the code transfer procedure of transmitting said IDL definition, the object implementation source, and the method source to a server. Said server program generation procedure The distributed object automatic generation method characterized by having the server generation procedure which compiles and links the procedure of receiving the transmitted code, and the received code, and generates an execute-form server program.
[Claim 8] It is the approach of generating automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation procedure and the server program generation procedure which generates a distributed object are included. Said client program generation procedure The object division procedure which extracts class processing from said source code, and distinguishes the activation location using a keyword, The global variable class generation procedure which generates the global variable deputy class which extracts a global variable from said source code and object division information, corresponds to the global variable

object and this which manage these IDL definition and these, and is prepared in a client, About the object judged by said division to be server activation, an IDL definition, The object implementation source and the server method source are created. While creating the deputy class source which corresponds to an object implementation and is prepared in a client side The non-extracting part and client activation class of the source of said user description language are used as a main source, and the processing to the server class of a main source is permuted by the access processing to said deputy class. The reference to a global variable, The main generation procedure which permutes substitution processing by the access processing to said global variable deputy class, and generates the client source, Said client source, the deputy class source, an IDL definition, a global variable deputy class, The client generation procedure which considers a global variable IDL definition as an input, compiles and links these, and generates an execute-form client program, It has the code transfer procedure of transmitting said IDL definition, the object implementation source, the method source, a global variable class, and its IDL definition to a server. Said server program generation procedure is a distributed object automatic generation method characterized by having the server generation procedure which compiles and links the procedure of receiving said transmitted code, and the received code, and generates an execute-form server program.

[Claim 9] Said client program generation procedure is claim 7 characterized by to include the processing which said deputy class changes into the format that it is called from the client source and an object request broker defines the set of a carrier beam argument, to generate said deputy class, and for said object implementation to include the processing which changes the set of a carrier beam argument into the format of a server method from an object request broker, and to generate said object implementation source, or a distributed object automatic generation method given in eight.

[Claim 10] Said client program generation procedure said global variable deputy class The processing changed into the format that it is called from the client source and an object request broker defines the set of a carrier beam argument is included. Said global variable deputy class is generated. Said global variable class The distributed object automatic generation method according to claim 8 characterized by including the processing which changes the set of a carrier beam argument into the format of a server method program from an object request broker, and generating said global variable class.

[Claim 11] They are claim 7 which said object division procedure is equipped with keyword information for every class of the corresponding to two or more kinds of user description languages, and is characterized by each keyword information having the logical expression between one or more keyword, and an activation tab-control-specification value at the time of logic enactment, or a distributed object automatic generation method given in eight.

[Claim 12] Said object division procedure is claim 7 characterized by to have the procedure decide an activation location about said extracted class processing according to the explicit activation tab-control-specification information first marked on the source of a user description language, and the procedure judge the activation location of the class processing without explicit activation tab control specification using said keyword information, or a distributed object automatic generation method given in eight.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention relates to the system and approach of generating automatically the distributed object of server activation, and the program of a client side only from the source code written especially with the user description language about the system and approach of generating a distributed object automatically.
[0002]
[Description of the Prior Art] When CORBA (Common Object Request Broker Architecture) which is the distributed object technology which united the client/server with object-oriented is conventionally used as a means to develop a distributed application, The extract of the object which operates on a server machine as a procedure of development, The definition of IDL (interface definition language) of the object which operates on a server machine, It is necessary to perform compile, a link, etc. starting of an IDL compiler and after that using the stub code and stub header which were generated, and to generate generation of the client program in a client machine, and the server program in a server machine.
[0003]
[Problem(s) to be Solved by the Invention] However, there were the following troubles in the above-mentioned conventional technique.
[0004] The 1st trouble is not easy for designing carving the object which operates on a server machine, and the object which operates only with a client machine, and I hear that it is difficult to change the operating environment of an object at the time of expansion, and it is in it.
[0005] The reason is for the need that a server and a client add a hand to a program to occur, when it is necessary to define IDL first and a hand is added to IDL.
[0006] The 2nd trouble is saying that a user needs to master advanced CORBA agreement and IDL linguistic knowledge like the object call which met conversion (it is called a MASHA ring below) to argument packing with which the user's needed to master IDL language in addition to the user description language to which are usually used, and met CORBA agreement from the user description language, and CORBA agreement.
[0007] The reason must describe logic to the program source outputted from the IDL compiler for server object development. Moreover, a client program needs to describe the call processing to the deputy object outputted from the IDL compiler. It is necessary to once make the argument at the time of a functional call with a user description language MASHA ring description in that case. Moreover, it is because the processing (it is called an AMMASHA ring below) which returns argument packing which met CORBA agreement to the form where he can understand a user description language is required for objection in a server program.
[0008] The 3rd trouble is being unable to change the active position of an object easily but changing the active position of an object with the definition of IDL, and a big manday burden called program modification.
[0009] The reason is for the activity shown on the 1st and 2nd troubles to occur, in order to change the active position of an object.
[0010] The 4th trouble is saying that describing without a mistake of the object which operates

between different machines on both sides of a CORBA communication link has complicated debugging easily.
[0011] Although X routine is immediately called when the object which described the reason in a single machine or single user language performs a routine call called X which the user described When the CORBA communication link is inserted, MASHA ring processing, connection with a server object, Since the user routine X can be gradually processed through a server object call and AMMASHA ring processing, mixing of a bug is altogether prevented for it not being easy and part specification when a bug mixes not being easy, either.
[0012] The 5th trouble is performing compile and a link according to an execute machine, and saying for a user that it is inconvenient, in order to create the object which operates between different machines on both sides of a CORBA communication link.
[0013] Since a distributed object is gradually completed through activation of IDL compile with a server machine, compile of a source code, and activation of a link after performing a transfer of the source code to a server machine, and a transfer of an IDL definition, the reason is to take time and effort far compared with the case where it describes in a single machine or single user language.
[0014] The 6th trouble's carrying out object division of the object which is performing the activity of referring to the global variable or a screen function conventionally is saying that it is a big manday burden to a user.
[0015] The reason is because a user needs to divide a client side function or a server side function each time, although the screen function is usually created on the assumption that the utilization from a client program.
[0016] Moreover, when dotted with refer to the global variable, it is necessary to devise adding a function call parameter etc. but, and although it is simple for a user, program correction impact is because it is comparatively large.
[0017]
[Means for Solving the Problem] The 1st distributed object automatic generative system by this invention It is the system which generates automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation means and a server program generation means to generate a distributed object are included. Said client program generation means An object division means to extract class processing from said source code, and to distinguish the activation location using a keyword, About the object judged to be server activation, an IDL definition, the object implementation source, The server method source is created. While generating the deputy class source which corresponds to an object implementation and is prepared in a client side A main generation means to use the non-extracting part and client activation class of the source of said user description language as a main source, to permute the processing to the server class of a main source by the access processing to said deputy class, and to generate the client source, A client generation means to consider said client source, the deputy class source, and an IDL definition as an input, to compile and link these, and to generate an execute-form client program, It has a code transfer means to transmit said IDL definition, the object implementation source, and the method source to a server. Said server program generation means It is characterized by having Cong Pal and a server generation means to link and to generate an execute-form server program for a means to receive the transmitted code, and the received code.
[0018] The 2nd distributed object automatic generative system by this invention It is the system which generates automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation means and a server program generation means to generate a distributed object are included. Said client program generation means An object division means to extract class processing from said source code, and to distinguish the activation location using a keyword, A global variable class generation means to generate the global variable deputy class which extracts a global variable from said source code and object division information, corresponds to the global variable object and this which manage these IDL definition and these,

and is prepared in a client, About the object judged by said division to be server activation, an IDL definition, The object implementation source and the server method source are created. While creating the deputy class source which corresponds to an object implementation and is prepared in a client side, the non-extracting part and client activation class of the source of said user description language are used as a main source. The processing to the server class of a main source is permuted by the access processing to said deputy class. A main generation means to permute the reference to a global variable, and substitution processing by the access processing to said global variable deputy class, and to generate the client source, Said client source, the deputy class source, an IDL definition, a global variable deputy class, A client generation means to consider a global variable IDL definition as an input, to compile and link these, and to generate an execute-form client program, It has a code transfer means to transmit said IDL definition, the object implementation source, the method source, a global variable class, and its IDL definition to a server. Said server program generation means is characterized by having a server generation means to compile and link a means to receive said transmitted code, and the received code, and to generate an execute-form server program.

[0019] The 3rd distributed object automatic generative system by this invention As for said client program generation means, said deputy class is called from the client source. The set of a carrier beam argument The processing changed into the format which an object request broker defines is included. Generate said deputy class and the processing said whose object implementation changes the set of a carrier beam argument into the format of a server method program from an object request broker is included. It is characterized by generating said object implementation source.

[0020] The 4th distributed object automatic generative system by this invention Said client program generation means said global variable deputy class The processing changed into the format that it is called from the client source and an object request broker defines the set of a carrier beam argument is included. Said global variable deputy class is generated, the processing said whose global variable class changes the set of a carrier beam argument into the format of a server method program from an object request broker is included, and it is characterized by generating said global variable class.

[0021] Said object division means is equipped with keyword information for every class of the by the 5th distributed object automatic generative system by this invention corresponding to two or more kinds of user description languages, and each keyword information is characterized by having the logical expression between one or more keyword, and an activation tab-control-specification value at the time of logic formation.

[0022] The 6th distributed object automatic generative system by this invention is characterized by for said object division means to have a means decide an activation location about said extracted class processing according to the explicit activation tab-control-specification information first marked on the source of a user description language, and a means judge the activation location of the class processing without explicit activation tab control specification using said keyword information.

[0023] The 1st distributed object automatic generation method by this invention It is the approach of generating automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation procedure and the server program generation procedure which generates a distributed object are included. Said client program generation procedure The object division procedure which extracts class processing from said source code, and distinguishes the activation location using a keyword, About the object judged to be server activation, an IDL definition, the object implementation source, The server method source is created. While generating the deputy class source which corresponds to an object implementation and is prepared in a client side The main generation procedure which uses the non-extracting part and client activation class of the source of said user description language as a main source, permutes the processing to the server class of a main source by the access processing to said deputy class, and generates the client source, The client generation procedure which considers said client source, the deputy class source, and an IDL definition as an input, compiles and links

these, and generates an execute-form client program, It has the code transfer procedure of transmitting said IDL definition, the object implementation source, and the method source to a server. Said server program generation procedure It is characterized by having the server generation procedure which compiles and links the procedure of receiving the transmitted code, and the received code, and generates an execute-form server program.
[0024] The 2nd distributed object automatic generation method by this invention It is the approach of generating automatically the distributed object of server activation, and the client program of a client side only from the source code written with the user description language. A client program generation procedure and the server program generation procedure which generates a distributed object are included. Said client program generation procedure The object division procedure which extracts class processing from said source code, and distinguishes the activation location using a keyword, The global variable class generation procedure which generates the global variable deputy class which extracts a global variable from said source code and object division information, corresponds to the global variable object and this which manage these IDL definition and these, and is prepared in a client, About the object judged by said division to be server activation, an IDL definition, The object implementation source and the server method source are created. While creating the deputy class source which corresponds to an object implementation and is prepared in a client side The non-extracting part and client activation class of the source of said user description language are used as a main source, and the processing to the server class of a main source is permuted by the access processing to said deputy class. The reference to a global variable, The main generation procedure which permutes substitution processing by the access processing to said global variable deputy class, and generates the client source, Said client source, the deputy class source, an IDL definition, a global variable deputy class, The client generation procedure which considers a global variable IDL definition as an input, compiles and links these, and generates an execute-form client program, It has the code transfer procedure of transmitting said IDL definition, the object implementation source, the method source, a global variable class, and its IDL definition to a server. Said server program generation procedure is characterized by having the server generation procedure which compiles and links the procedure of receiving said transmitted code, and the received code, and generates an execute-form server program.
[0025] The 3rd distributed object automatic generation method by this invention As for said client program generation procedure, said deputy class is called from the client source. The set of a carrier beam argument The processing changed into the format which an object request broker defines is included. Said deputy class is generated, the processing from which said object implementation changes the set of a carrier beam argument into the format of a server method from an object request broker is included, and it is characterized by generating said object implementation source.
[0026] The 4th distributed object automatic generation method by this invention Said client program generation procedure said global variable deputy class The processing changed into the format that it is called from the client source and an object request broker defines the set of a carrier beam argument is included. Said global variable deputy class is generated, said global variable class includes the processing which changes the set of a carrier beam argument into the format of a server method program from an object request broker, and it is characterized by generating said global variable class.
[0027] Said object division procedure is equipped with keyword information for every class of the by the 5th distributed object automatic generation method by this invention corresponding to two or more kinds of user description languages, and each keyword information is characterized by having the logical expression between one or more keyword, and an activation tab-control-specification value at the time of logic enactment.
[0028] The 6th distributed object automatic generation method by this invention is characterized by for said object division procedure to have the procedure decide an activation location about said extracted class processing according to the explicit activation tab-control-specification information first marked on the source of a user description language, and the procedure judge the activation location of the class processing without explicit activation tab control specification

using said keyword information.
[0029]
[Embodiment of the Invention] The distributed object automatic generative system and approach by language functional interpretation of this invention The program described with the user description language which can operate on a single machine is considered as an input. Perform the semantic analysis and a client object and a server object are judged automatically. To a client program side, the MASHA ring processing to a client deputy object, Connection processing and server method call processing (it is hereafter called deputy object control description) are generated automatically. A server object side is provided with the configuration which can generate automatically AMMASHA ring processing and method call processing (it is hereafter called a server implementation).
[0030] Moreover, in that case, what a user has to manage is the program described with the user description language which can operate on a single machine, and an IDL definition, deputy object control description, and a server implementation offer management needlessness and a possible configuration.
[0031] Moreover, this configuration offers the configuration which does not remain in only changing the source program described with the user description language, but generates automatically the formal client program which can be performed to up to a client machine, and generates a distributed object automatically also to up to a server machine as a format which can be performed.
[0032] Moreover, an object arrangement location is made switchable in an instant by enabling a user to specify clearly whether a function is arranged to a server, or it arranges to a client by shaking a specific alphabetic character, a notation, or WORD on the source of a user description language.
[0033] Moreover, the automatic judging of the activation location of class processing is enabled by the keyword which made the source of a user description language applicable to retrieval.
[0034] Next, the gestalt of operation of this invention is explained with reference to a drawing. With reference to drawing 1 , the distributed object automatic generative system by the language functional interpretation of this invention and one example of an approach operate by program control.
[0035] A client machine 100 and server machines 200 and 201 are connected in the network 300, a client machine 100 has the client program generation section 1, and server machines 200 and 201 have the server program generation section 3.
[0036] The client program generation section 1 is a means to generate the client program 2 which performs the demand to a distributed object. The server program generation section 3 is a means to generate the server distributed objects 4 and 5 which receive the demand from a client program 2.
[0037] Drawing 2 is the block diagram having shown the configurations and these contents of I/O of the client program generation section 1 and the server program generation section 3.
[0038] The client program generation section 1 has the object division section 10, the main generation section 11, the global variable class generation section 12, the code transfer section 13, and the client generation section 14.
[0039] The server program generation section 3 has the receive section of the code transfer section 13, and the server generation section 30.
[0040] The object division section 10 determines and divides the object which considers the user description language source file 6 as an input, and is arranged to a server, and the object arranged to a client. Therefore, corresponding to two or more kinds of user description languages, it has the keyword definition 101 for every class of the, and each keyword definition has two or more sets of these by making into a unit information which consists of an activation tab-control-specification value at the time of the logical expression between one or more keyword, and logic formation (server-client).
[0041] The main generation section 11 generates automatically the client source 23 and the server method source 25.
[0042] In accordance with the content of the object arranged to a server, the object

implementation 26 which manages the deputy object source 24 with deputy object control description and a server implementation is generated automatically in that case. The IDL definition 27 for connecting between distributed objects is generated automatically.
[0043] The global variable class generation section 12 generates automatically the global variable class and the IDL definition 28 which put together only the deputy object control description to a global variable, and a server object implementation.
[0044] The client generation section 14 considers the client source 23, the deputy object source 24, the IDL definition 27, a global variable class, and the IDL definition 28 as an input, and generates a client program 2 automatically.
[0045] a code -- a transfer -- the section -- 13 -- a server -- a program -- generating -- a sake -- being required -- a file -- a server -- transmitting -- this -- a receive section -- a server -- generation -- the section -- 30 -- receiving -- a server -- a method -- the source -- 25 -- ' -- an object -- an implementation -- 26 -- ' -- IDL -- a definition -- 27 -- ' -- a global variable -- a class -- and -- IDL -- a definition -- 28 -- ' -- inputting .
[0046] The server generation section 30 generates automatically the server distributed object 4 and a global variable (server) 5.
[0047] Thus, it makes it possible to generate a client program and a server program automatically from the user description language which can operate on a single machine.
[0048] Next, actuation of this whole operation gestalt is explained to a detail with reference to the flow chart of drawing 3 .
[0049] The object division section 10 reads the user description language source file 6, performs language syntax analysis and a semantic analysis, extracts a class, and performs a class division in a server class and a client class about the thing in which server actuation is possible.
[0050] The main generation section 11 creates each file of the IDL definition 27 of drawing 2 , a global variable class and the IDL definition 28, the object implementation 26, the deputy object source 24, the server method source 25, and the client source 23. (Step 1) .
[0051] The code transfer section 13 transmits each file of the server MESEDDO source 25 of drawing 2 required in order to generate a server object, the object implementation 26, the IDL definition 27, a global variable class, and the IDL definition 28 to a server (step 2).
[0052] a code -- a transfer -- the section -- 13 -- a receive section -- a server -- generation -- the section -- 30 -- a server -- a method -- the source -- 25 -- ' -- an object -- an implementation -- 26 -- ' -- IDL -- a definition -- 27 -- ' -- a global variable -- a class -- and -- IDL -- a definition -- 28 -- ' -- generation of a delivery server program -- requiring (step 3) .

[0053] The client generation section 14 considers the client source 23, the deputy object source 24, the IDL definition 27, a global variable class, and the IDL definition 28 as an input, compiles these, and generates a client program 2 automatically (step 4).
[0054] Next, it explains using an example. Especially, processing of step 1 which is the description of this application is explained in detail.
[0055] As shown in drawing 4 , the user description language source file 6 is read first (step 1-1). next, a server and a client -- it judges automatically whether it is the object description which should be located in which (step 1-2). The thing of class processing level is extracted, an object mounting position when the user is doing activation tab control specification clearly is determined, and the class without assignment of a user carries out processing which what cannot be arranged to a language function top server is taken up, and is made into a client class as drawing 5 shows the detail of step 1-2.
[0056] For example, a user's explicit activation tab control specification is specified by the comment field of the user description language source. A server, a client, and those without assignment are recognized in the alphabetic character (S/C/tooth space) next to the initiation definition statement characters (! etc.) of a comment.
[0057] About a class without assignment, it considers as a client class according to the keyword definition table 101 of a description language response. If it is specified as the client by the definition table by "OUT" and AND formation of "DISPLAY", the class which fills these will be taken as a client class.

[0058] If the extract of a server object is completed at return and step 1-2 to drawing 4 , the main generation section 11 generates the IDL definition to a server object automatically (step 1-3). In a server object, the server object implementation which takes charge of the AMMASHA ring between IDL language and a user description language is generated automatically (step 1-4).

[0059] Next, the deputy class of which it makes a show as if a server object exists in a client is generated automatically (step 1-5). A deputy class takes charge of the MASHA ring between a user description language and IDL language.
[0060] The detail of step 1-7 to 1-9 which the global variable class generation section 12 performs is as being shown in drawing 6 , extracts a global variable from the user description language source and its division information, and generates automatically the server object which manages a global variable. A global variable has also from a server the appearance which can be referred to transparent also from a client, the deputy class which takes charge of a MASHA ring, and the global variable server implementation which takes charge of an AMMASHA ring.
[0061] At step 1-10, processing to the user description language source is permuted. As drawing 7 shows, the code which performs refer to the global variable and value setting out to a global variable is extracted, and automatic conversion of the applicable part is carried out at the processing description to a global variable deputy object.
[0062] At step 1-6, automatic conversion of the access processing to the object located in a server is carried out at the access processing to a deputy object.
[0063] In S1-11, initial processing and a post process required for CORBA distributed object activation are inserted in a client main source. Moreover, a global variable object inserts the processing which activates the server object of a global variable within initial processing so that it may become available from immediately after activation.
[0064] Thus, compile and a link are performed automatically and each generated file will be in the condition which can be performed immediately, as drawing 3 mentioned above shows.
[0065] The relation of each file at the time of activation is shown in drawing 9 . R1 of drawing 9 is the example of the Maine processing of a client program. the initial processing generated automatically inserts in the Maine processing -- having -- **** -- initial processing of CORBA, R2, and .. piece activation of the global variable management object is carried out as processing to R5 shows.
[0066] When referring to [G1] the global variable occurs in a client program, a value includes the global variable management object located in a server by work of a global variable deputy object [finishing / conversion] by drawing 7 . Moreover, R6 client class which cannot be located or located in a server is performed on a client machine by the usual language function of a user description language.
 [0067] Access processing to the server object generated so that it might be automatically located in a server by this invention is performed by the server through processing of R7-R11. The need of the processing result of R11 that return and a user are conscious of existence of a server object on a source code will be lost to R7 through R10-R8.
[0068] Only an object-oriented mold source file is not targetted for the distributed object automatic generative system and approach of this invention as a user description language source file 6, and they are aimed at all description languages. For example, C, COBOL, Java, etc. are mentioned.
[0069] Moreover, the distributed object automatic generative system and approach of this invention are aimed not at invention which specialized in CORBA but at all distributed object bases. For example, it is applicable to DCE (distributed computing environment), DCOM (distributed component object model), etc.
[0070]
[Effect of the Invention] It has the 1st effectiveness referred to as that the distributed object automatic generative system and approach of this invention do not have the need of newly [in order / which was explained above / to carry out the semantic interpretation of the user description language and to generate a distributed object and a client program automatically] in addition to the user description language to which a user is usually used mastering IDL language like.

[0071] Moreover, since it has come out to exclude the source code transfer for the compile and the link by the user etc., it has the 2nd effectiveness that implementation of an IDL language function, compile, and a link can be thoroughly concealed from a user.
[0072] The distributed object automatic generative system and approach of this invention generate a server object and a client object according to the mark on a user description language source file, and since they perform automatically the addition of the extract of a global variable, a MASHA ring, and AMMASHA ring processing, they have the 3rd effectiveness of saying the active position of an object that a user can change easily.
[0073] That is, there is processing required in order to usually describe a distributed object plentifully, in order to enable it to operate the object which operates by the client conventionally by the server, it carries out an IDL definition, and although the user itself was performing the MASHA ring and the AMMASHA ring, it does these activities unnecessary.
[0074] Moreover, although the object activation location needed to be changed having applied big manday, the manday burden can be removed by this invention to compare prototype creation time and the activation engine performance.
[0075] Since the distributed object automatic generative system and approach of this invention generate a distributed object automatically, without adding a hand to the source code described in user language, the input file of ******** can be created as a program which operates on the usual single machine, and a user has the 4th effectiveness that the debugger to which are used can be used.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] The block diagram showing the whole example configuration of an operation gestalt of the distributed object automatic generative system of this invention, and an approach.
[Drawing 2] The block diagram having shown the configurations and these contents of I/O of the client program generation section 1 of drawing 1 , and the server program generation section 3.
[Drawing 3] The flow chart which shows the actuation by the whole example of an operation gestalt of the distributed object automatic generative system of this invention, and shows the whole distributed object automatic generation method step of this invention.
[Drawing 4] The flow chart which shows actuation of the client program generation section 1, and the detail step of a client program generation procedure.
[Drawing 5] The flow chart which shows the processing step of object division.
[Drawing 6] The flow chart which shows a global variable class generation processing step.
[Drawing 7] Drawing showing the example of concrete processing of step 1-10 of drawing 4 .
[Drawing 8] Drawing showing the example of concrete processing of step 1-6 of drawing 4 .
[Drawing 9] Drawing showing each activation module of the program generated by the client and the server, and the situation at the time of activation.
[Description of Notations]
1 Client Program Generation Section
10 Object Division Section
101 Keyword Definition
11 The Main Generation Section
12 Global Variable Class Generation Section
13 Code Transfer Section
14 Client Generation Section
2 Client Program
23 Client Source
24 Deputy Object Source
25 Server Method Source
26 Object Implementation
27 IDL Definition
28 Global Variable Class and IDL Definition
3 Server Program Generation Section
30 Server Generation Section
4 Server Distributed Object
6 User Description Language Source File
100 Client Machine
200 201 Server machine
300 Network

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－132502
（P2002－132502A）
(43)公開日　平成14年５月10日(2002.5.10)

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)	
G06F	9/44		G06F	9/44	530P	5B045
		530		15/16	620T	5B076
	9/45				630A	5B081
	15/16	620		9/06	620A	
		630		9/44	322A	

審査請求　有　請求項の数12　ＯＬ　（全 12 頁）

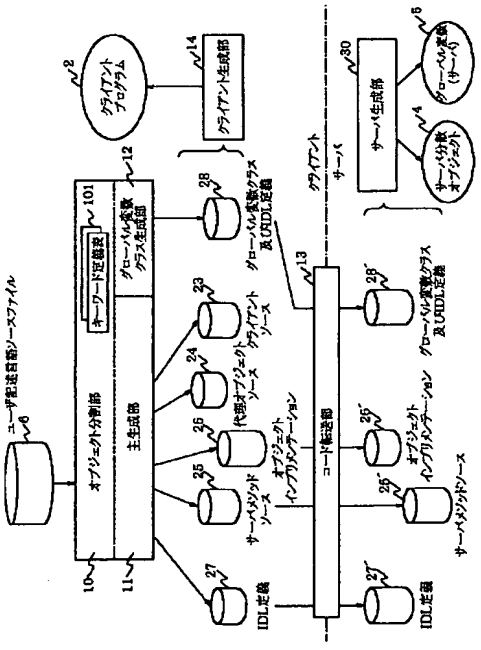
(21)出願番号　特願2000－325652(P2000－325652)
(22)出願日　平成12年10月25日(2000.10.25)

(71)出願人　000242666
北陸日本電気ソフトウェア株式会社
石川県石川郡鶴来町安養寺１番地
(72)発明者　能田　昌彦
石川県石川郡鶴来町安養寺１番地　北陸日本電気ソフトウェア株式会社内
(74)代理人　100082935
弁理士　京本　直樹　（外２名）

Ｆターム(参考)　5B045 BB11 BB28 BB47 GG09
5B076 DD04
5B081 CC41

(54)【発明の名称】　言語機能解釈による分散オブジェクト自動生成システム及び方法

(57)【要約】
【課題】　従来、ユーザ記述言語のソースから分散処理プログラムを作成する為に、ユーザが行っていた実行位置決定、サーバオブジェクトのＩＤＬ記述、マーシャリング、アンマーシャリング処理コーディング等を不要にする。
【解決手段】　クライアントプログラム生成部１は、ユーザ記述言語ソース６からクラスを抽出し実行位置を判別するオブジェクト分割部１０、サーバオブジェクトについてＩＤＬ定義２７、インプリメンテーション２６、メソッド２５、代理オブジェクト２４を生成し、ソース６のサーバオブジェクト以外の部分のサーバへのアクセスを代理に置換しクライアントソース２３を生成する主生成部１１と、ソース２３、代理オブジェクト２４、ＩＤＬ定義２７をコンパイルリンクするクライアント生成部１４と、コード転送部１３とを有し、サーバプログラム生成部３は、受信したソースをコンパイルリンクするサーバ生成手段とを有す。



1

【特許請求の範囲】

【請求項１】　ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成するシステムであって、クライアントプログラム生成手段と分散オブジェクトを生成するサーバプログラム生成手段とを含み、前記クライアントプログラム生成手段は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手段と、サーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを生成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスをメインソースとしメインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換しクライアントソースを生成する主生成手段と、前記クライアントソース、代理クラスソース、ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手段と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとをサーバに転送するコード転送手段とを有し、前記サーバプログラム生成手段は、転送されたコードを受信する手段と受信したコードをコンバル及びリンクして実行形式サーバプログラムを生成するサーバ生成手段とを有すことを特徴とする分散オブジェクト自動生成システム。

【請求項２】　ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成するシステムであって、クライアントプログラム生成手段と分散オブジェクトを生成するサーバプログラム生成手段とを含み、前記クライアントプログラム生成手段は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手段と、前記ソースコード及びオブジェクト分割情報からグローバル変数を抽出しこれらのＩＤＬ定義とこれらを管理するグローバル変数オブジェクトとこれに対応しクライアントに設けるグローバル変数代理クラスとを生成するグローバル変数クラス生成手段と、前記分割でサーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを作成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスとをメインソースとし、メインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換し、グローバル変数への参照、代入処理を前記グローバル変数代理クラスへのアクセス処理に置換してクライアントソースを生成する主生成手段と、前記クライアントソース、代理クラスソース、ＩＤＬ定義、グローバル変数代理クラス、グローバル変数ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手段と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとグローバル変数クラスとそのＩＤＬ定義とをサーバに転送するコード転送手段とを有し、前記サーバプログラム生成手段は、前記転送されたコードを受信する手段と受信したコードをコンパイル及びリンクして実行形式サーバプログラムを生成するサーバ生成手段とを有すことを特徴とする分散オブジェクト自動生成システム。

【請求項３】　前記クライアントプログラム生成手段は、前記代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませて、前記代理クラスを生成し、前記オブジェクトインプリメンテーションがオブジェクトリクエストブローカから受けた引数のセットをサーバメソッドプログラムの形式に変換する処理を含ませて、前記オブジェクトインプリメンテーションソースを生成することを特徴とする請求項１、又は２記載の分散オブジェクト自動生成システム。

【請求項４】　前記クライアントプログラム生成手段は、前記グローバル変数代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませ、前記グローバル変数代理クラスを生成し、前記グローバル変数クラスがオブジェクトリクエストブローカから受けた引数のセットをサーバメソッドプログラムの形式に変換する処理を含ませ、前記グローバル変数クラスを生成することを特徴とする請求項２記載の分散オブジェクト自動生成システム。

【請求項５】　前記オブジェクト分割手段は複数種類のユーザ記述言語に対応して、その種類毎にキーワード情報を備え、各キーワード情報は、１個以上のキーワードとキーワード間の論理式と論理成立時の実行位置指定値とを有すことを特徴とする請求項１、又は２記載の分散オブジェクト自動生成システム。

【請求項６】　前記オブジェクト分割手段は、前記抽出したクラス処理について、先ずユーザ記述言語のソース上にマークされた明示的な実行位置指定情報に従って実行位置を決める手段と、明示的な実行位置指定が無いクラス処理の実行位置を前記キーワード情報により判定する手段とを有すことを特徴とする請求項１、又は２記載の分散オブジェクト自動生成システム。

【請求項７】　ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成する方法であって、クライアントプログラム生成手順と分散オブジェクトを生成するサーバプログラム生成手順とを含み、前記

2

クライアントプログラム生成手順は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手順と、サーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを生成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスとをメインソースとしメインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換しクライアントソースを生成する主生成手順と、前記クライアントソース、代理クラスソース、ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手順と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとをサーバに転送するコード転送手順とを有し、前記サーバプログラム生成手順は、転送されたコードを受信する手順と受信したコードをコンパイル及びリンクして実行形式サーバプログラムを生成するサーバ生成手順とを有すことを特徴とする分散オブジェクト自動生成方法。

【請求項８】　ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成する方法であって、クライアントプログラム生成手順と分散オブジェクトを生成するサーバプログラム生成手順とを含み、前記クライアントプログラム生成手順は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手順と、前記ソースコード及びオブジェクト分割情報からグローバル変数を抽出しこれらのＩＤＬ定義とこれらを管理するグローバル変数オブジェクトとこれに対応しクライアントに設けるグローバル変数代理クラスとを生成するグローバル変数クラス生成手順と、前記分割でサーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを作成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスとをメインソースとしメインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換しグローバル変数への参照、代入処理を前記グローバル変数代理クラスへのアクセス処理に置換してクライアントソースを生成する主生成手順と、前記クライアントソース、代理クラスソース、ＩＤＬ定義、グローバル変数代理クラス、グローバル変数ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手順と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとグローバル変数クラスとそのＩＤＬ

定義とをサーバに転送するコード転送手順とを有し、前記サーバプログラム生成手順は、前記転送されたコードを受信する手順と受信したコードをコンパイル及びリンクして実行形式サーバプログラムを生成するサーバ生成手順とを有すことを特徴とする分散オブジェクト自動生成方法。

【請求項９】　前記クライアントプログラム生成手順は、前記代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませて、前記代理クラスを生成し、前記オブジェクトインプリメンテーションがオブジェクトリクエストブローカから受けた引数のセットをサーバメソッドの形式に変換する処理を含ませて、前記オブジェクトインプリメンテーションソースを生成することを特徴とする請求項７、又は８記載の分散オブジェクト自動生成方法。

【請求項１０】　前記クライアントプログラム生成手順は、前記グローバル変数代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませ、前記グローバル変数代理クラスを生成し、前記グローバル変数クラスが、オブジェクトリクエストブローカから受けた引数のセットをサーバメソッドプログラムの形式に変換する処理を含ませ、前記グローバル変数クラスを生成することを特徴とする請求項８記載の分散オブジェクト自動生成方法。

【請求項１１】　前記オブジェクト分割手順は複数種類のユーザ記述言語に対応して、その種類毎にキーワード情報を備え、各キーワード情報は、１個以上のキーワードとキーワード間の論理式と論理成立時の実行位置指定値とを有すことを特徴とする請求項７、又は８記載の分散オブジェクト自動生成方法。

【請求項１２】　前記オブジェクト分割手順は、前記抽出したクラス処理について、先ずユーザ記述言語のソース上にマークされた明示的な実行位置指定情報に従って実行位置を決める手順と、明示的な実行位置指定が無いクラス処理の実行位置を前記キーワード情報により判定する手順とを有すことを特徴とする請求項７、又は８記載の分散オブジェクト自動生成方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は分散オブジェクトを自動生成するシステム及び方法に関し、特にユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のプログラムを自動生成するシステムおよび方法に関する。

【０００２】

【従来の技術】従来より、分散アプリケーションを開発する手段として、オブジェクト指向とクライアント／サーバを融合した分散オブジェクト技術であるＣＯＲＢＡ

（Ｃｏｍｍｏｎ　Ｏｂｊｅｃｔ　Ｒｅｑｕｅｓｔ　Ｂｒｏｋｅｒ　Ａｒｃｈｉｔｅｃｔｕｒｅ)を用いる場合、開発の手順としてはサーバマシン上で動作するオブジェクトの抽出、サーバマシン上で動作するオブジェクトのＩＤＬ（インタフェース定義言語）の定義、ＩＤＬコンパイラの起動、その後、生成されたスタブコードとスタブヘッダを用いてコンパイルやリンク等を行い、クライアントマシンでのクライアントプログラムの生成、およびサーバマシンでのサーバプログラムの生成を行う必要がある。

【０００３】

【発明が解決しようとする課題】しかし、上記従来技術には、次の様な問題点があった。

【０００４】第１の問題点は、サーバマシン上で動作するオブジェクトとクライアントマシンでのみ動作するオブジェクトを切り分けながら設計することは容易ではなく、又、機能拡張時にオブジェクトの動作環境を変更することが困難であるということである。

【０００５】その理由は、最初にＩＤＬを定義する必要があり、ＩＤＬに手を加えた場合サーバ、クライアント共にプログラムに手を加える必要が発生する為である。

【０００６】第２の問題点は、利用者が、普段使い慣れたユーザ記述言語以外にＩＤＬ言語を習得する必要があり、又、ユーザ記述言語からＣＯＲＢＡ規約にそった引数バッキングへの変換（以下マーシャリングと呼ぶ）、ＣＯＲＢＡ規約にそったオブジェクト呼び出しの様に、利用者が高度なＣＯＲＢＡ規約やＩＤＬ言語知識を習得する必要があると言うことである。

【０００７】その理由は、サーバオブジェクト開発の為にはＩＤＬコンパイラから出力されたプログラムソースにロジックを記述せねばならず、又クライアントプログラムはＩＤＬコンパイラから出力された代理オブジェクトに対する呼び出し処理を記述する必要があり、その際ユーザ記述言語での機能呼び出し時の引数を一旦マーシャリング記述にする必要があり、又、反対にサーバプログラムではＣＯＲＢＡ規約にそった引数バッキングをユーザ記述言語が理解出来る形に戻す処理（以下アンマーシャリングと呼ぶ）が必要である為である。

【０００８】第３の問題点は、オブジェクトの動作位置を容易に変更出来ず、オブジェクトの動作位置を変更するにはＩＤＬの定義、プログラムの修正という大きな工数負担となることである。

【０００９】その理由は、オブジェクトの動作位置を変更する為には第１及び第２の問題点で示した作業が発生する為である。

【００１０】第４の問題点は、ＣＯＲＢＡ通信を挟んで異なるマシン間で動作するオブジェクトをミスなく記述することは容易ではなく、又デバッグが複雑であると言うことである。

【００１１】その理由は、単一マシンや単一ユーザ言語で記述したオブジェクトはユーザが記述したＸというルーチン呼び出しを行った場合、即座にＸルーチンが呼び出されるが、ＣＯＲＢＡ通信を挟んでいる場合には、マーシャリング処理、サーバオブジェクトへの接続、サーバオブジェクト呼び出し、アンマーシャリング処理を経て漸く利用者ルーチンＸの処理を行うことが出来る為、バグの混入を未然に全て防ぐのは容易ではなく、バグが混入した場合の箇所特定も容易ではない為である。

【００１２】第５の問題点は、ＣＯＲＢＡ通信を挟んで異なるマシン間で動作するオブジェクトを作成するには、実行マシンに合わせてコンパイル及びリンクを行う必要があり、利用者にとって不便であると言うことである。

【００１３】その理由は、サーバマシンへのソースコードの転送、ＩＤＬ定義の転送を行った後、サーバマシンでのＩＤＬコンパイルの実行、ソースコードのコンパイル及びリンクの実行を経て漸く分散オブジェクトが完成する為、単一マシンや単一ユーザ言語で記述した場合に比べはるかに手間がかかる為である。

【００１４】第６の問題点は、従来、グローバル変数参照や画面機能の使用を行っているオブジェクトをオブジェクト分割することは利用者への大きな工数負担となっていると言うことである。

【００１５】その理由は、画面機能はクライアントプログラムからの利用を前提とし作成されているのが通常であるが、利用者がその都度クライアント側機能かサーバ側機能かを分割する必要がある為である。

【００１６】又、グローバル変数参照が点在する場合には関数呼び出しパラメータを追加するなどの工夫をする必要があるが、利用者にとっては単純だが、プログラム修正インパクトは、比較的大きい為である。

【００１７】

【課題を解決するための手段】本発明による第１の分散オブジェクト自動生成システムは、ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成するシステムであって、クライアントプログラム生成手段と分散オブジェクトを生成するサーバプログラム生成手段とを含み、前記クライアントプログラム生成手段は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手段と、サーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを生成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスをメインソースとしメインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換しクライアントソースを生成する主生成手段と、前記クライアントソース、

代理クラスソース、ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手段と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとをサーバに転送するコード転送手段とを有し、前記サーバプログラム生成手段は、転送されたコードを受信する手段と受信したコードをコンパル及びリンクして実行形式サーバプログラムを生成するサーバ生成手段とを有すことを特徴とする。

【００１８】本発明による第２の分散オブジェクト自動生成システムは、ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成するシステムであって、クライアントプログラム生成手段と分散オブジェクトを生成するサーバプログラム生成手段とを含み、前記クライアントプログラム生成手段は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手段と、前記ソースコード及びオブジェクト分割情報からグローバル変数を抽出しこれらのＩＤＬ定義とこれらを管理するグローバル変数オブジェクトとこれに対応しクライアントに設けるグローバル変数代理クラスとを生成するグローバル変数クラス生成手段と、前記分割でサーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを作成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスとをメインソースとし、メインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換し、グローバル変数への参照、代入処理を前記グローバル変数代理クラスへのアクセス処理に置換してクライアントソースを生成する主生成手段と、前記クライアントソース、代理クラスソース、ＩＤＬ定義、グローバル変数代理クラス、グローバル変数ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手段と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとグローバル変数クラスとそのＩＤＬ定義とをサーバに転送するコード転送手段とを有し、前記サーバプログラム生成手段は、前記転送されたコードを受信する手段と受信したコードをコンパイル及びリンクして実行形式サーバプログラムを生成するサーバ生成手段とを有すことを特徴とする。

【００１９】本発明による第３の分散オブジェクト自動生成システムは、前記クライアントプログラム生成手段は、前記代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませて、前記代理クラスを生成し、前記オブジェクトインプリメンテーションがオブジェクトリクエストブローカから受けた引数のセットをサーバメソッドプログラムの形式に変換する処理を含ませて、前記オブジェクトインプリメンテーションソースを生成することを特徴とする。

【００２０】本発明による第４の分散オブジェクト自動生成システムは、前記クライアントプログラム生成手段は、前記グローバル変数代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませ、前記グローバル変数代理クラスを生成し、前記グローバル変数クラスがオブジェクトリクエストブローカから受けた引数のセットをサーバメソッドプログラムの形式に変換する処理を含ませ、前記グローバル変数クラスを生成することを特徴とする。

【００２１】本発明による第５の分散オブジェクト自動生成システムは、前記オブジェクト分割手段は複数種類のユーザ記述言語に対応して、その種類毎にキーワード情報を備え、各キーワード情報は、１個以上のキーワードとキーワード間の論理式と論理成立時の実行位置指定値とを有すことを特徴とする。

【００２２】本発明による第６の分散オブジェクト自動生成システムは、前記オブジェクト分割手段は、前記抽出したクラス処理について、先ずユーザ記述言語のソース上にマークされた明示的な実行位置指定情報に従って実行位置を決める手段と、明示的な実行位置指定が無いクラス処理の実行位置を前記キーワード情報により判定する手段とを有すことを特徴とする。

【００２３】本発明による第１の分散オブジェクト自動生成方法は、ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成する方法であって、クライアントプログラム生成手順と分散オブジェクトを生成するサーバプログラム生成手順とを含み、前記クライアントプログラム生成手順は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手順と、サーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを生成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスとをメインソースとしメインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換しクライアントソースを生成する主生成手順と、前記クライアントソース、代理クラスソース、ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手順と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとをサーバに転送するコード転送手順とを有し、前記サーバプログラム

生成手順は、転送されたコードを受信する手順と受信したコードをコンパイル及びリンクして実行形式サーバプログラムを生成するサーバ生成手順とを有すことを特徴とする。

【００２４】本発明による第２の分散オブジェクト自動生成方法は、ユーザ記述言語で書かれたソースコードのみからサーバ実行の分散オブジェクトとクライアント側のクライアントプログラムを自動生成する方法であって、クライアントプログラム生成手順と分散オブジェクトを生成するサーバプログラム生成手順とを含み、前記クライアントプログラム生成手順は、前記ソースコードからクラス処理を抽出しその実行位置をキーワードを用い判別するオブジェクト分割手順と、前記ソースコード及びオブジェクト分割情報からグローバル変数を抽出しこれらのＩＤＬ定義とこれらを管理するグローバル変数オブジェクトとこれに対応しクライアントに設けるグローバル変数代理クラスとを生成するグローバル変数クラス生成手順と、前記分割でサーバ実行と判定されたオブジェクトについてＩＤＬ定義、オブジェクトインプリメンテーションソース、サーバメソッドソースを作成し、オブジェクトインプリメンテーションに対応しクライアント側に設ける代理クラスソースを作成すると共に前記ユーザ記述言語のソースの非抽出部分とクライアント実行クラスとをメインソースとしメインソースのサーバクラスへの処理を前記代理クラスへのアクセス処理に置換しグローバル変数への参照、代入処理を前記グローバル変数代理クラスへのアクセス処理に置換してクライアントソースを生成する主生成手順と、前記クライアントソース、代理クラスソース、ＩＤＬ定義、グローバル変数代理クラス、グローバル変数ＩＤＬ定義を入力とし、これらをコンパイル及びリンクして実行形式クライアントプログラムを生成するクライアント生成手順と、前記ＩＤＬ定義とオブジェクトインプリメンテーションソースとメソッドソースとグローバル変数クラスとそのＩＤＬ定義とをサーバに転送するコード転送手順とを有し、前記サーバプログラム生成手順は、前記転送されたコードを受信する手順と受信したコードをコンパイル及びリンクして実行形式サーバプログラムを生成するサーバ生成手順とを有すことを特徴とする。

【００２５】本発明による第３の分散オブジェクト自動生成方法は、前記クライアントプログラム生成手順は、前記代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませて、前記代理クラスを生成し、前記オブジェクトインプリメンテーションがオブジェクトリクエストブローカから受けた引数のセットをサーバメソッドの形式に変換する処理を含ませて、前記オブジェクトインプリメンテーションソースを生成することを特徴とする。

【００２６】本発明による第４の分散オブジェクト自動生成方法は、前記クライアントプログラム生成手順は、前記グローバル変数代理クラスが、クライアントソースから呼び出され受けた引数のセットを、オブジェクトリクエストブローカの定める形式に変換する処理を含ませ、前記グローバル変数代理クラスを生成し、前記グローバル変数クラスが、オブジェクトリクエストブローカから受けた引数のセットをサーバメソッドプログラムの形式に変換する処理を含ませ、前記グローバル変数クラスを生成することを特徴とする。

【００２７】本発明による第５の分散オブジェクト自動生成方法は、前記オブジェクト分割手順は複数種類のユーザ記述言語に対応して、その種類毎にキーワード情報を備え、各キーワード情報は、１個以上のキーワードとキーワード間の論理式と論理成立時の実行位置指定値とを有すことを特徴とする。

【００２８】本発明による第６の分散オブジェクト自動生成方法は、前記オブジェクト分割手順は、前記抽出したクラス処理について、先ずユーザ記述言語のソース上にマークされた明示的な実行位置指定情報に従って実行位置を決める手順と、明示的な実行位置指定が無いクラス処理の実行位置を前記キーワード情報により判定する手順とを有すことを特徴とする。

【００２９】

【発明の実施の形態】本発明の言語機能解釈による分散オブジェクト自動生成システム及び方法は、単一マシン上で動作可能なユーザ記述言語で記述されたプログラムを入力とし、その意味解析を行い、自動的にクライアントオブジェクトとサーバオブジェクトとを判断し、クライアントプログラム側にはクライアント代理オブジェクトへのマーシャリング処理、接続処理、サーバメソッド呼び出し処理（以下、代理オブジェクト制御記述と呼ぶ）を自動生成し、サーバオブジェクト側にはアンマーシャリング処理、メソッド呼び出し処理（以下、サーバインプリメンテーションと呼ぶ）を自動生成出来る構成を提供するものである。

【００３０】又、その際、利用者が管理しなければいけないものは、単一マシン上で動作可能なユーザ記述言語で記述されたプログラムであり、ＩＤＬ定義や代理オブジェクト制御記述及びサーバインプリメンテーションは管理不要と出来る構成を提供するものである。

【００３１】又、本構成は、ユーザ記述言語で記述されたソースプログラムを単に変換することにとどまらず、クライアントマシン上へ実行可能形式クライアントプログラムを自動生成し、サーバマシン上へも分散オブジェクトを実行可能形式として自動生成する構成を提供するものである。

【００３２】又、ユーザ記述言語のソース上に特定の文字、記号、或いはワードを振ることによって、機能をサーバに配置するか、クライアントに配置するかを利用者が明示的に指定することを可能とすることにより、オブ

ジェクト配置位置を瞬時に切り替え可能とする。

【００３３】又、ユーザ記述言語のソースを検索対象としたキーワードによりクラス処理の実行位置の自動判定を可能とする。

【００３４】次に、本発明の実施の形態について図面を参照し説明する。図１を参照し、本発明の言語機能解釈による分散オブジェクト自動生成システム及び方法の一実施例は、プログラム制御により動作する。

【００３５】クライアントマシン１００とサーバマシン２００、２０１はネットワーク３００で接続されており、クライアントマシン１００はクライアントプログラム生成部１を有し、サーバマシン２００、２０１は、サーバプログラム生成部３を有する。

【００３６】クライアントプログラム生成部１は、分散オブジェクトへの要求を行うクライアントプログラム２を生成する手段である。サーバプログラム生成部３は、クライアントプログラム２からの要求を受け付けるサーバ分散オブジェクト４、５を生成する手段である。

【００３７】図２はクライアントプログラム生成部１、サーバプログラム生成部３の構成及びこれらの入出力内容を示したブロック図である。

【００３８】クライアントプログラム生成部１は、オブジェクト分割部１０、主生成部１１、グローバル変数クラス生成部１２、コード転送部１３、クライアント生成部１４を有す。

【００３９】サーバプログラム生成部３は、コード転送部１３の受信部とサーバ生成部３０を有す。

【００４０】オブジェクト分割部１０は、ユーザ記述言語ソースファイル６を入力としサーバに配置するオブジェクトとクライアントに配置するオブジェクトを決定・分割する。　その為に、複数種類のユーザ記述言語に対応して、その種類毎にキーワード定義１０１を備え、各キーワード定義は、１個以上のキーワードとキーワード間の論理式と論理成立時の実行位置指定値（サーバ／クライアント）からなる情報を単位としてこれらを複数組有す。

【００４１】主生成部１１は、クライアントソース２３とサーバメソッドソース２５を自動生成する。

【００４２】その際、サーバに配置するオブジェクトの内容に沿って、代理オブジェクト制御記述を持つ代理オブジェクトソース２４とサーバインプリメンテーションを司るオブジェクトインプリメンテーション２６を自動生成する。分散オブジェクト間を結ぶ為のＩＤＬ定義２７を自動生成する。

【００４３】グローバル変数クラス生成部１２は、グローバル変数への代理オブジェクト制御記述、サーバオブジェクトインプリメンテーションだけを一纏めにした、グローバル変数クラス及びＩＤＬ定義２８を自動生成する。

【００４４】クライアント生成部１４は、クライアントソース２３、代理オブジェクトソース２４、ＩＤＬ定義２７、グローバル変数クラス及びＩＤＬ定義２８を入力とし、クライアントプログラム２を自動生成する。

【００４５】コード転送部１３は、サーバプログラムを生成する為に必要なファイルをサーバに転送し、この受信部はサーバ生成部３０に対して、サーバメソッドソース２５’、オブジェクトインプリメンテーション２６’、ＩＤＬ定義２７’、グローバル変数クラス及びＩＤＬ定義２８’を入力する。

【００４６】サーバ生成部３０は、サーバ分散オブジェクト４とグローバル変数（サーバ）５を自動生成する。

【００４７】この様にして、単一マシン上で動作可能なユーザ記述言語からクライアントプログラムやサーバプログラムを自動生成することを可能にする。

【００４８】次に、図３のフローチャートを参照して本実施形態の全体の動作について詳細に説明する。

【００４９】オブジェクト分割部１０は、ユーザ記述言語ソースファイル６を読み込み、言語構文解析・意味解析を行い、クラスを抽出し、サーバ動作可能なものについてサーバクラス、クライアントクラスにクラス分けを行なう。

【００５０】主生成部１１は図２のＩＤＬ定義２７、グローバル変数クラス及びＩＤＬ定義２８、オブジェクトインプリメンテーション２６、代理オブジェクトソース２４、サーバメソッドソース２５、クライアントソース２３の各ファイルを作成する。（ステップ１）。

【００５１】コード転送部１３は、サーバオブジェクトを生成する為に必要な図２のサーバメセッドソース２５、オブジェクトインプリメンテーション２６、ＩＤＬ定義２７、グローバル変数クラス及びＩＤＬ定義２８の各ファイルをサーバに転送する（ステップ２）。

【００５２】コード転送部１３の受信部は、サーバ生成部３０にサーバメソッドソース２５’、オブジェクトインプリメンテーション２６’、ＩＤＬ定義２７’、グローバル変数クラス及びＩＤＬ定義２８’を渡しサーバプログラムの生成を要求する（ステップ３）。

【００５３】クライアント生成部１４はクライアントソース２３、代理オブジェクトソース２４、ＩＤＬ定義２７、グローバル変数クラス及びＩＤＬ定義２８を入力とし、これらをコンパイルしクライアントプログラム２を自動生成する（ステップ４）。

【００５４】次に、具体例を用いて説明する。特に、本願の特徴であるステップ１の処理を詳しく説明する。

【００５５】図４に示す様に、先ずユーザ記述言語ソースファイル６を読み込む（ステップ１－１）。次にサーバ、クライアントどちらに位置すべきオブジェクト記述かを自動的に判断する（ステップ１－２）。ステップ１－２の詳細は図５で示す通り、クラス処理レベルのものを抽出し、、利用者が明示的に実行位置指定をしている場合のオブジェクト実装位置の決定を行ない、利用者の

指定のないクラスは言語機能上サーバに配置出来ないものをピックアップしクライアントクラスとする処理をする。

【００５６】例えば、利用者の明示的な実行位置指定は、ユーザ記述言語ソースのコメント欄で指定されている。コメントの開始定義文字（！等）の次の文字（Ｓ／Ｃ／スペース）でサーバ、クライアント、指定無しを見分ける。

【００５７】指定無しのクラスについては、記述言語対応のキーワード定義表１０１に従ってクライアントクラスとする。定義表に「ＯＵＴ」と、「ＤＩＳＰＬＡＹ」のＡＮＤ成立でクライアントと指定されていればこれらを満たすクラスはクライアントクラスとする。

【００５８】図４に戻り、ステップ１－２でサーバオブジェクトの抽出が完了すると、主生成部１１はサーバオブジェクトに対するＩＤＬ定義を自動生成する（ステップ１－３）。サーバオブジェクトにおいて、ＩＤＬ言語とユーザ記述言語間のアンマーシャリングを担当するサーバオブジェクトインプリメンテーションを自動生成する（ステップ１－４）。

【００５９】次に、サーバオブジェクトがあたかもクライアントに存在するかの様に見せかける代理クラスを自動生成する（ステップ１－５）。代理クラスは、ユーザ記述言語とＩＤＬ言語間のマーシャリングを担当する。

【００６０】グローバル変数クラス生成部１２が行うステップ１－７～１－９の詳細は図６に示す通りで、ユーザ記述言語ソースと、その分割情報からグローバル変数を抽出し、グローバル変数を管理するサーバオブジェクトを自動生成する。グローバル変数はサーバからもクライアントからも透過的に参照出来る様、マーシャリングを担当する代理クラスと、アンマーシャリングを担当するグローバル変数サーバインプリメンテーションを持つ。

【００６１】ステップ１－１０では、ユーザ記述言語ソースに対する処理の置換を行う。図７で示す様に、グローバル変数参照や、グローバル変数への値設定を行うコードを抽出し、該当部分をグローバル変数代理オブジェクトへの処理記述に自動変換する。

【００６２】ステップ１－６では、サーバに位置するオブジェクトへのアクセス処理を代理オブジェクトへのアクセス処理に自動変換する。

【００６３】Ｓ１－１１では、クライアントメインソースにＣＯＲＢＡ分散オブジェクト実行に必要な初期処理や終了処理が挿入される。又、グローバル変数オブジェクトは実行直後から利用可能となる様に、初期処理内でグローバル変数のサーバオブジェクトを活性化させる処理を挿入する。

【００６４】この様にして生成された各々のファイルは、前述した図３で示す様に、自動的にコンパイル及びリンクが行われ、すぐに実行が可能な状態となる。

【００６５】実行時の各々のファイルの関係を図９に示す。図９のＲ１はクライアントプログラムのメイン処理の例である。メイン処理には自動生成された初期処理が挿入されており、ＣＯＲＢＡ初期処理、Ｒ２、・・Ｒ５までの処理で示す通りグローバル変数管理オブジェクトが一個活性化される。

【００６６】クライアントプログラムでグローバル変数参照Ｇ１が発生した時、図７で変換済みのグローバル変数代理オブジェクトの働きにより、サーバに位置するグローバル変数管理オブジェクトに値が渡る。又、サーバに位置しない又は位置出来ないＲ６クライアントクラスはユーザ記述言語の通常の言語機能により、クライアントマシン上で実行する。

【００６７】本発明により自動的にサーバに位置する様に生成されたサーバオブジェクトへのアクセス処理は、Ｒ７～Ｒ１１の処理を経て、サーバで実行される。Ｒ１１の処理結果はＲ１０～Ｒ８を経てＲ７へ戻り、利用者はソースコード上、サーバオブジェクトの存在を意識する必要が無くなることとなる。

【００６８】本発明の分散オブジェクト自動生成システム及び方法は、ユーザ記述言語ソースファイル６としてオブジェクト指向型ソースファイルのみを対象とするものでなく、あらゆる記述言語を対象とする。例えば、Ｃ言語、ＣＯＢＯＬ、Ｊａｖａなどが挙げられる。

【００６９】又、本発明の分散オブジェクト自動生成システム及び方法は、ＣＯＲＢＡに特化した発明ではなく、あらゆる分散オブジェクト基盤を対象とする。例えばＤＣＥ（ｄｉｓｔｒｉｂｕｔｅｄ ｃｏｍｐｕｔｉｎｇ ｅｎｖｉｒｏｎｍｅｎｔ）、ＤＣＯＭ（ｄｉｓｔｒｉｂｕｔｅｄ ｃｏｍｐｏｎｅｎｔ ｏｂｊｅｃｔ ｍｏｄｅｌ）等にも適用出来る。

【００７０】

【発明の効果】以上説明した様に、本発明の分散オブジェクト自動生成システム及び方法は、ユーザ記述言語を意味解釈し、分散オブジェクト、クライアントプログラムを自動的に生成する為、利用者が普段使い慣れたユーザ記述言語以外に新たにＩＤＬ言語を習得する必要が無いと言う第１の効果を有す。

【００７１】又、ユーザによるコンパイル及びリンクの為のソースコード転送なども省くことが出きる為、利用者からＩＤＬ言語機能やコンパイル及びリンクの実施を完全に隠蔽出来るという第２の効果を有す。

【００７２】本発明の分散オブジェクト自動生成システム及び方法は、ユーザ記述言語ソースファイル上のマークに従ってサーバオブジェクト、クライアントオブジェクトを生成し、グローバル変数の抽出、マーシャリング、アンマーシャリング処理の追加を自動的に行うのでオブジェクトの動作位置を利用者が簡単に変更出来ると言う第３の効果を有す。

【００７３】即ち、通常分散オブジェクトを記述する為

15

に必要な処理は多々あり、従来クライアントで動作するオブジェクトをサーバで動作出来る様にする為にはＩＤＬ定義をし、マーシャリング、アンマーシャリングを利用者自身が行なっていたがこれら作業を不要とする。

【００７４】又、プロトタイプ作成時や実行性能を比較したい場合に、大きな工数をかけてオブジェクト実行位置を変更する必要があったが本発明により、その工数負担を取り除くことが出来る。

【００７５】本発明の分散オブジェクト自動生成システム及び方法は、利用者言語で記述されたソースコードに手を加えることなく分散オブジェクトを自動生成するので、本発明のの入力ファイルは通常の単一マシン上で動作するプログラムとして作成出来、利用者は使い慣れたデバッガを利用出来るという第４の効果を有す。

【図面の簡単な説明】

【図１】本発明の分散オブジェクト自動生成システム及び方法の実施形態例の全体構成を示すブロック図。

【図２】図１のクライアントプログラム生成部１、サーバプログラム生成部３の構成及びこれらの入出力内容を示したブロック図。

【図３】本発明の分散オブジェクト自動生成システムの実施形態例の全体動作を示し、又本発明の分散オブジェクト自動生成方法の全体ステップを示すフローチャート。

【図４】クライアントプログラム生成部１の動作、クライアントプログラム生成手順の詳細ステップを示すフローチャート。

【図５】オブジェクト分割の処理ステップを示すフローチャート。

16

＊【図６】グローバル変数クラス生成処理ステップを示すフローチャート。

【図７】図４のステップ１－１０の具体処理例を示す図。

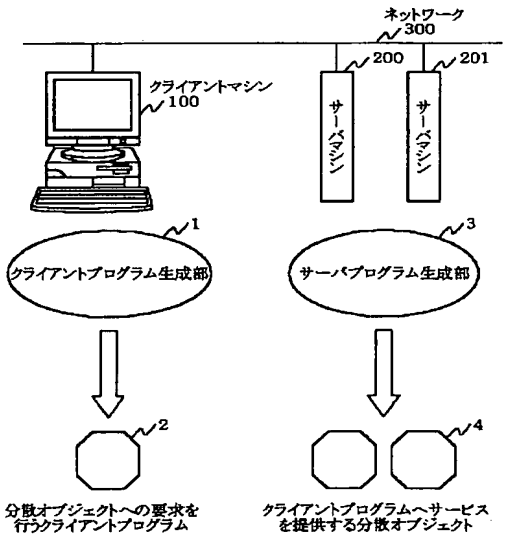
【図８】図４のステップ１－６の具体処理例を示す図。

【図９】クライアント、サーバに生成されたプログラムの各実行モジュールと実行時の様子を示す図。

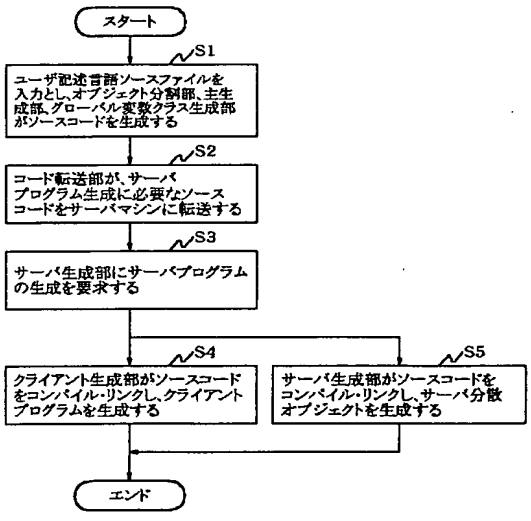
【符号の説明】

１　クライアントプログラム生成部
１０　オブジェクト分割部
１０１　キーワード定義
１１　主生成部
１２　グローバル変数クラス生成部
１３　コード転送部
１４　クライアント生成部
２　クライアントプログラム
２３　クライアントソース
２４　代理オブジェクトソース
２５　サーバメソッドソース
２６　オブジェクトインプリメンテーション
２７　ＩＤＬ定義
２８　グローバル変数クラス及びＩＤＬ定義
３　サーバプログラム生成部
３０　サーバ生成部
４　サーバ分散オブジェクト
６　ユーザ記述言語ソースファイル
１００　クライアントマシン
２００、２０１　サーバマシン
＊　３００　ネットワーク

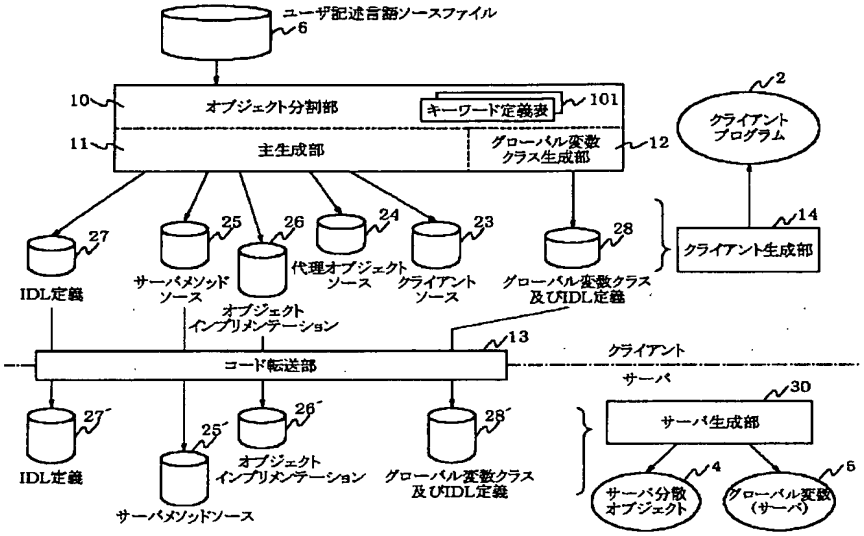
【図１】



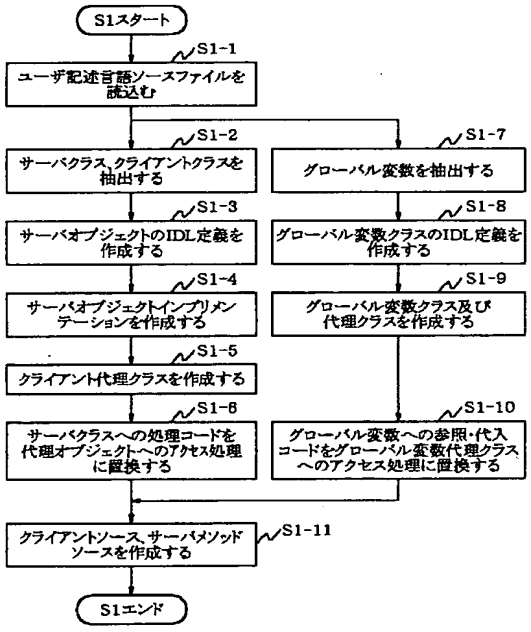
【図３】



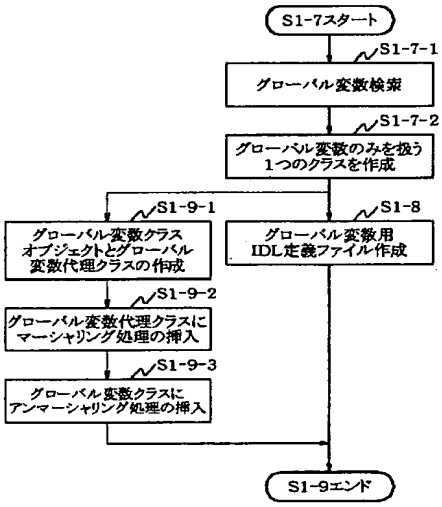
【図２】


ユーザ記述言語ソースファイル
6
10
オブジェクト分割部
キーワード定義表
101
11
主生成部
グローバル変数クラス生成部
12
2
クライアントプログラム
14
クライアント生成部
27
IDL定義
25
サーバメソッドソース
26
代理オブジェクトソース
24
クライアントソース
23
28
グローバル変数クラス及びIDL定義
オブジェクトインプリメンテーション
13
コード転送部
クライアント
サーバ
30
サーバ生成部
27´
IDL定義
25´
サーバメソッドソース
26´
オブジェクトインプリメンテーション
28´
グローバル変数クラス及びIDL定義
4
サーバ分散オブジェクト
6
グローバル変数(サーバ)

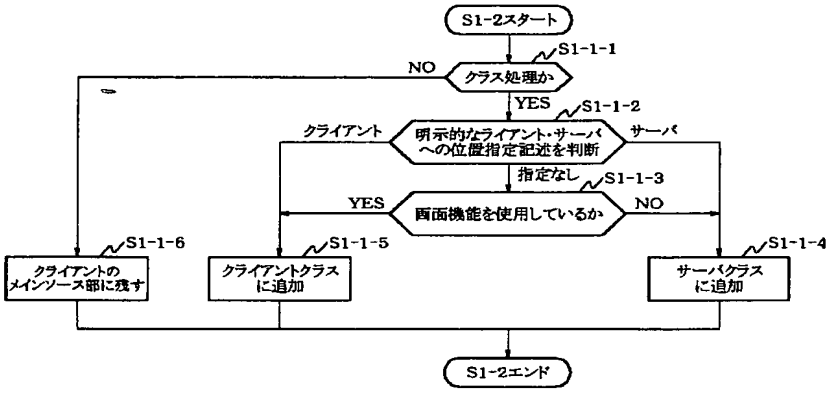
【図４】


S1スタート
S1-1
ユーザ記述言語ソースファイルを読込む
S1-2
サーバクラス、クライアントクラスを抽出する
S1-3
サーバオブジェクトのIDL定義を作成する
S1-4
サーバオブジェクトインプリメンテーションを作成する
S1-5
クライアント代理クラスを作成する
S1-6
サーバクラスへの処理コードを代理オブジェクトへのアクセス処理に置換する
S1-7
グローバル変数を抽出する
S1-8
グローバル変数クラスのIDL定義を作成する
S1-9
グローバル変数クラス及び代理クラスを作成する
S1-10
グローバル変数への参照・代入コードをグローバル変数代理クラスへのアクセス処理に置換する
S1-11
クライアントソース、サーバメソッドソースを作成する
S1エンド

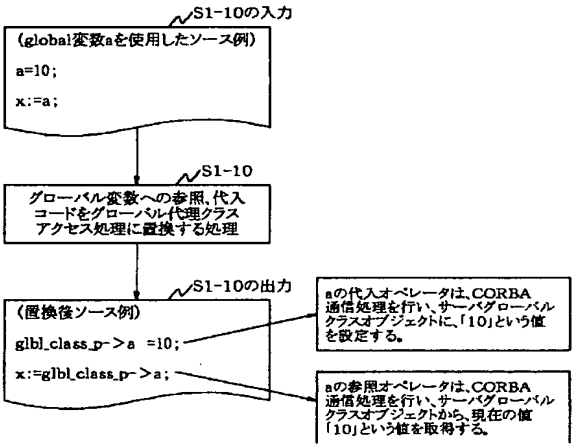
【図６】


S1-7スタート
S1-7-1
グローバル変数検索
S1-7-2
グローバル変数のみを扱う1つのクラスを作成
S1-9-1
グローバル変数クラスオブジェクトとグローバル変数代理クラスの作成
S1-8
グローバル変数用IDL定義ファイル作成
S1-9-2
グローバル変数代理クラスにマーシャリング処理の挿入
S1-9-3
グローバル変数クラスにアンマーシャリング処理の挿入
S1-9エンド

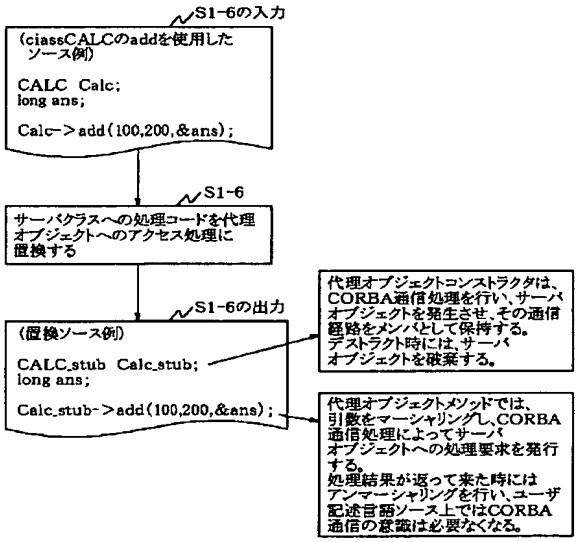
【図５】


S1-2スタート
S1-1-1
NO
クラス処理か
YES
S1-1-2
クライアント
明示的なクライアント・サーバへの位置指定記述を判断
サーバ
指定なし
S1-1-3
YES
画面機能を使用しているか
NO
S1-1-6
クライアントのメインソース部に残す
S1-1-5
クライアントクラスに追加
S1-1-4
サーバクラスに追加
S1-2エンド

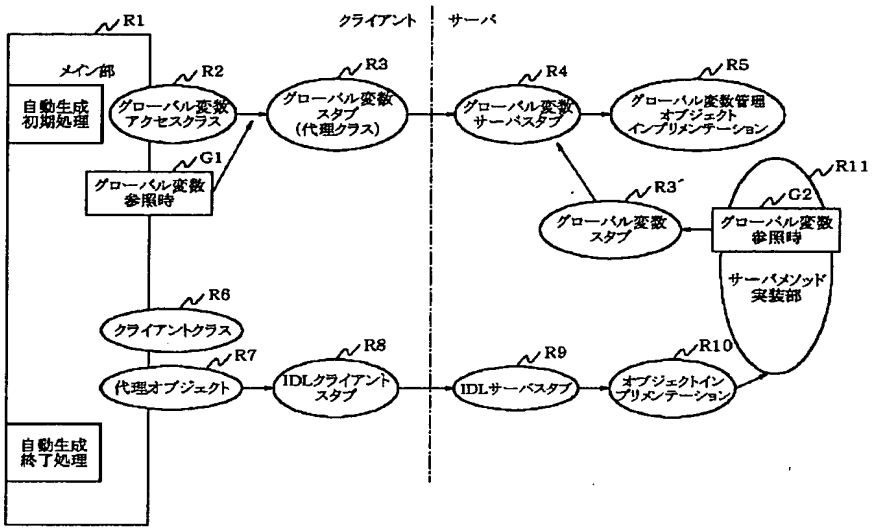
【図７】


S1-10の入力
(global変数aを使用したソース例)
a=10;
x:=a;
S1-10
グローバル変数への参照,代入コードをグローバル代理クラスアクセス処理に置換する処理
S1-10の出力
(置換後ソース例)
glbl_class_p->a =10;
x:=glbl_class_p->a;
aの代入オペレータは、CORBA通信処理を行い、サーバグローバルクラスオブジェクトに、「10」という値を設定する。
aの参照オペレータは、CORBA通信処理を行い、サーバグローバルクラスオブジェクトから、現在の値「10」という値を取得する。

【図８】


S1-6の入力
(ciassCALCのaddを使用したソース例)
CALC Calc;
long ans;
Calc->add(100,200,&ans);
S1-6
サーバクラスへの処理コードを代理オブジェクトへのアクセス処理に置換する
S1-6の出力
(置換ソース例)
CALC_stub Calc_stub;
long ans;
Calc_stub->add(100,200,&ans);
代理オブジェクトコンストラクタは、CORBA通信処理を行い、サーバオブジェクトを発生させ、その通信経路をメンバとして保持する。
デストラクト時には、サーバオブジェクトを破棄する。
代理オブジェクトメソッドでは、引数をマーシャリングし、CORBA通信処理によってサーバオブジェクトへの処理要求を発行する。
処理結果が返って来た時にはアンマーシャリングを行い、ユーザ記述言語ソース上ではCORBA通信の意識は必要なくなる。

【図９】


R1
クライアント
サーバ
メイン部
自動生成
初期処理
R2
グローバル変数
アクセスクラス
R3
グローバル変数
スタブ
(代理クラス)
R4
グローバル変数
サーバスタブ
R5
グローバル変数管理
オブジェクト
インプリメンテーション
G1
グローバル変数
参照時
R3'
グローバル変数
スタブ
R11
G2
グローバル変数
参照時
サーバメソッド
実装部
R6
クライアントクラス
R7
代理オブジェクト
R8
IDLクライアント
スタブ
R9
IDLサーバスタブ
R10
オブジェクトイン
プリメンテーション
自動生成
終了処理